Echo Suppression

Reinhold Brückner
(*1963, Konstanz, RBR2018@gmx.com)

November 2025

Application of simple high school math to an
(almost) everyday problem of no importance.

When having a skype conversation or participating in a podcast without headset, one encounters a nuisance: one hears one's own voice from the microphone of the conversation partners.
You might wonder why this problem hardly occurs when you set your telephone in loudspeaker mode. The reason is that, on a telephone, loudspeaker (earphone) and microphone are geometrically arranged and constructed in such a way that "they cannot hear each other".
This is generally not the case in podcasts and the like. The microphone stands in the room and accepts sound from any source around, including the loudspeaker.
In order to eliminate this "own" signal from the contribution of the partner one first has to know how long it takes to get from the loudspeaker to the microphone.
This depends on the particular situation and has to be evaluated in the beginning of the conversation or whenever microphone or loudspeaker are moved.
This can be quite a pain in the neck if the conversation partner moves around with the microphone yet keeps the receiving loudspeaker fixed.
In that case, distance has to be evaluated continuously.
"Distance" means, of course, delay between the incoming signal (IN) and outgoing signal (OUT).

The method to do all this is an elementary application of high school vector calculation.
Let's start with notation: (we always use the IN signal as example)
Column vectors are denoted $| IN >$
Row vectors are denoted $< IN |$
The squared norm of a vector is its scalar product with itself, that is row vector times column vector (in this order)

$$< IN | IN > = \sum IN_k \, IN_k$$

Where IN_k is the vector's k-th component and we sum over k.

We call a vector *normalized* when its norm is 1. We denote it by an index N (eg $| IN_N >$)

This can always be achieved by dividing the original, unnormalized vector by the square root of its squared norm, calculated by the prescription above.

$$| IN_N > = | IN > / \sqrt{< IN | IN >}$$

Practically, we start by breaking up the OUT signal into intervals of 2 milliseconds length.
We use a sampling rate of 50 kHz, meaning that we will have 100 amplitude values in our (OUT).
This represents a vector with 100 dimensions. That's a little more than the 2 or 3 dimensions encountered in school but the principle is the same.
The IN signal to be eliminated is also sectioned in 2 ms intervals, again with 100 dimensions.

Elimination is achieved by subtracting the IN vector-component contained in the OUT-vector from the OUT vector.
The resulting "cleaned" OUT signal vector is then (C index for "cleaned")

$$| OUT_C (t) > = | OUT(t) > - | IN_N (t-\Delta t) >< IN_N(t-\Delta t) | OUT(t) >$$

Notice that we have to use the *normalized* IN-vector.

We also have to use an IN signal from a former time $(t - \Delta t)$ because of the delay to reach the microphone.

In order to find the right delay, we scan over a range of "plausible" delay times Δt and form its squared scalar product with the *normalized* OUT vector

$$(< IN_N (t-\Delta t) \ | OUT_N (t) >)^2$$

This gives a positive number. Its maximum (as a function of Δt) gives the desired delay.

The length of sampling gives the minimal delay between the actual OUT signal and its "cleaned" version to be sent.
In a fixed microphone-loudspeaker setup IN-OUT delay is evaluated only at the start.
The cleaning calculation is so simple that it will not introduce any noticable additional delay.

However, in a moving setup, IN-OUT delay has to be continuously adjusted which might well add an additional one millisecond delay.
This would give a total delay of 3 milliseconds.

Does this matter ?
Not at all. Sound takes 3 milliseconds for one yard which is a small conversation distance.
The match between video image and sound will be considered completely normal by our brains.

echo suppression

It might occur that the room has an echo and that we hear ourselves twice. In that case, we can proceed by first eliminating the direct voice input and then eliminate the echo from the "precleaned" output signal.
The echo will be an IN-signal, but from a *former time* than the direct IN-signal.
Hence, we are dealing with two different vectors.
Does the order of "cleaning", direct first, echo second or the inverse, matter?
To find out, let's write it out.
To ease notation, we will denote the normalized *direct* IN-signal vector as | 1 >
and the normalized *echo* IN-signal as | 2 >
The cleaned OUT-signal will be denoted $| OUT_c >$
and the "double-cleaned OUT-signal as $| OUT_{cc} >$

The cleaned OUT-signal is given as above

$$| OUT_c > = | OUT > - | 1 >< 1 | OUT >$$

The double-cleaned OUT-signal

$$| OUT_{cc} > = | OUT_c > - | 2 >< 2 | OUT_c >$$

$$| OUT_{cc} > = | OUT > - | 1 >< 1 | OUT > - | 2 >< 2 | OUT > + | 2 >< 2 | 1 >< 1 | OUT >$$

The order turns out to be non-interchangeable since we do not get the same when we swap 1 and 2.

Let us write it slightly differently

$$| OUT_{cc} > = | OUT > - | 1 >< 1 | OUT > - | 2 >(< 2 | OUT > - < 2 | 1 >< 1 | OUT >)$$

If IN-signal vectors 1, 2 are largely orthogonal, meaning, they have nothing to do with each other, the last term will basically be zero and we can interchange the order of cleaning.

But this might not always be the case. Imagine some background music playing with a rythm. The rythm being repetitive, the echo can well coincide with the direct IN-signal.
Our second cleaning is then useless, since we already eliminated this rythm in the first cleaning.
This is born out in the formalism since this will weaken or cancel the second action. The term in the bracket becomes zero (or small).

This comes a little as a surprise and should certainly spur some deeper investigation.

Other applications

Among possible candidates for this formalism one might think of color manipulations.
Color space (for each pixel) is three-dimensional, white being then be a vector (1,1,1).
If we want to eliminate a color partly, we use the formalism as
(The index c stands now for "corrected")

$| C_c > = | C > - a| 1 >< 1 | C >$

The factor a is a number smaller than 1 ($a < 1$) and indicates the degree of desired elimination.
If $a = 1$, our remaining color space is only 2 - dimensional.

Note that this has nothing to do with green-screening. There, one just checks whether it's green or not and accordingly sets the color to zero or leaves it unscathed.

On this occasion: why do we have three fundamental colors?
Because we have three cones.
Had we four we would have four fundamental colors.